GUIDE DU SERVEUR

Service Support
Développeurs

Commodore

Chers développeurs,

Le serveur a été mis en place sur un Amiga 2500/30 fonctionnant sous UNIX système V.3.2. Nous vous invitons à découvrir les services qu'il offre aux développeurs commerciaux, certifiés et enregistrés. C'est votre ligne directe avec Commodore, ce n'est pas un concurrent d'autres serveurs. Vous pouvez le consulter avec un minitel 1B, ou un modem en interactif ou en batch.

La connexion en interactif (on-line) est simple à mettre en oeuvre, mais elle est nettement plus coûteuse que la connexion batch qui transmet régulièrement les fichiers sans intervention de l'utilisateur.

C'est un lieu de contact : les développeurs, comme les membres de l'équipe support de Commodore France ont tous une boîte aux lettres. Il est possible de joindre l'ensemble des développeurs européens et du personnel de Commodore depuis le serveur français. Sur le plan technique, on peut utiliser le programme elm (plus pratique que mailx) pour les connexions on-line mais il est vivement recommandé d'utiliser le logiciel AmigaUUCP pour transférer son courrier du serveur à un Amiga et vice-versa.

C'est un lieu de référence : une petite bibliothèque de programmes du domaine public et d'informations trop longues pour tenir dans les messages a été constituée. Vous pouvez y voir notamment, les fichiers des bulletins AmigaMail du support américain. Mais il y a surtout les dernières versions des logiciels systèmes de Commodore.

C'est un lieu d'information : la totalité des informations concernant l'Amiga circulant sur le réseau USENET est disponible dans le sous-système des news. On y trouve des discussions sur les domaines d'applications où l'Amiga excelle, le développement, les problèmes Hardware ou encore des sources et des exemples. Il y a trois groupes de discussions à suivre :

comp.sys.amiga.tech	pour le développement
comp.sources.amiga	pour les sources
adsp.*	pour toutes les discussions entre les développeurs européens.

La quantité d'informations qui circule est importante, même si parfois des questions idiotes encombrent le réseau.

Si vous ne disposez pas de logiciel de comunication adapté, le service technique peut vous faire parvenir une disquette où sont rassemblés des programmes utilitaires du domaine public.

Comment accéder à ce serveur ? C'est simple :

- si vous êtes muni d'un minitel 1B (80 colonnes), appuyez sur les touches FNCT et T puis sur la touche A pour passer en 80 colonnes, composez le numéro (1) 46 44 30 74, lorsque le modem du minitel accroche la porteuse, établissez la connexion.

- si vous êtes muni d'un minitel (tous types), d'un Amiga-Minitel (type Flam), lancez le programme d'émulation de terminal vt100 sur l'Amiga, composez le numéro (1) 46 44 30 74, lorsque le modem du minitel accroche la porteuse, établissez la connexion (le minitel comme les vt100 fonctionnent à 1200 bauds, sept bits de données et parité paire).

- si vous êtes muni d'un modem 1200/1200 et d'un ordinateur (tous types), lancez un programme d'émulation de terminal vt100 sur l'ordinateur, paramétrez-le pour fonctionner à 1200 bauds en 8 bits de données, et pas de parité, faites lui composer le numéro (1) 46 44 35 14 (par exemple en passant la commande HAYES +++ATDEP1,46443514).

Lorsque la connexion téléphonique est établie entre les deux modems, il est parfois nécessaire de taper au moins un retour chariot pour que le système Unix détecte l'appel. Vous devez alors voir apparaître la question :

login:

à laquelle il vous faut répondre par votre nom de connexion,

le système affiche alors la question :

password:

à laquelle il vous faudra répondre votre mot de passe.

Pour des raisons de sécurité, ce mot de passe n'est pas affiché.

Si vous avez fait une faute de frappe dans le nom ou dans le mot de passe, le système vous demandera les deux informations. Si vous voyez apparaître au lieu de login: et de password:, des caractères inidentifiables, vérifiez les paramètres de la configuration et le numéro d'appel. Si les troubles persistent, vous devez téléphoner à Francis Poulain ou à Nicolas Costes pour vérifier avec eux l'ensemble de la procédure.

ATTENTION : Unix fait la différence entre les caractères majuscules et minuscules, en règle générale, veillez à ne travailler qu'en minuscule. Toutefois si les premiers caractères reçus par le système Unix sont des majuscules, alors toute la transaction se fera en majuscule comme avec un terminal doté d'un jeu de caractère très réduit. Pour ne pas être confronté à ce problème, vous pouvez, si vous obtenez un login en majucule, envoyer un "CTRL D" au serveur ; celui-ci recommencera une session, mais cette fois, en minuscule.

D'autre part, cette version d'Unix ne gère pas les accents, veillez à ne taper que des caractères simples, ou à utiliser le programme MODIF pour convertir les caractères accentués.

Nicolas COSTES

Francis POULAIN

Généralités sur UNIX.

1) Description sommaire d'UNIX...

ou: Comment résumer 7 manuels de 200 pages en quelques lignes !

- Il est bon de se rappeler qu' UNIX a été écrit en C; ainsi, les minuscules et les majuscules sont interprétées différemment; par exemple: /users n'est pas le même répertoire que /Users

- La grande majorité des commandes possèdent des options (ex: ls -l); celles ci commencent toujours par le signe "-".

-UNIX est un système multitâche ET multi-utilisateur; il y a donc des systèmes de protection des données; c'est pourquoi, à chaque fichier UNIX sont attachés divers attributs :

- un propriétaire
- un groupe (celui du créateur du fichier)
- des droits d'accès (pour le propriétaire, pour le groupe et pour les autres)
- etc...(taille, dates, nombre de liens...)

Les droits d'accès sont lisibles grace a l'option -l de la commande ls

Exemple:

instruction:	ls -l toto						
résultat:	-rwxr-xr-x	1	francis	technic	118	Aug 23 17:49	toto
explication:	(1)	(2)	(3)	(4)	(5)	(6)	(7)

(1) droits d'accès: r:lecture
w:écriture
x:exécution
ici: francis: r+w+x
technic: x+r
autres: x

(2) nombre de liens
(3) propriétaire du fichier toto
(4) groupe auquel appartient le propriétaire
(5) taille en blocs (1 bloc= 512 octets)
(6) dernière date de modification
(7) nom du fichier

NB: Vous pouvez changer les droits d'accès de VOS fichiers en utilisant la commande "chmod".

2) Commandes UNIX

Voici une liste de commandes UNIX de base:

NB: les commandes Amiga correspondantes (même approximativement) sont mises entre parenthèses.

COMMANDE		DESCRIPTION	USAGE
cal		calendrier	cal
cat	(type)	impression	cat toto
cd	(cd)	change de répertoire	cd /tele
cd		va dans le répert. $HOME	cd
cmp		compare 2 fichiers	cmp toto titi
cp	(copy)	copie un fichier	cp titi tata
date	(date)	donne la date (selon TZ)	date
echo	(echo)	imprime	echo "bonjour"
find		sélectionne fichiers	
ls	(list/dir)	list le contenu d'un répert.	ls /users
	-l	"format long"	ls -l /users
	-ali		ls -ali
mkdir	(makedir)	crée un répertoire	mkdir essai
more	(more)	impression avec formatage	more toto
mv	(rename)	déplace ou renomme un fichier	mv toto titi
ps	(status)	donne états de vos processus	ps
	-edf	donne états des processus	ps -edf
pwd	(cd)	donne répertoire courant	pwd
rm	(delete)	détruit fichier	rm toto
rmdir	(delete)	détruit répertoire vide	rmdir essai
sh	(newcli)	lance un shell	sh
stty		donne paramètres du terminal	stty
	-a	idem avec toutes les options	stty -a
tty		donne nom du terminal	tty
who		liste des utilisateurs	who
	-am i	"qui êtes-vous ?"	who am i

3) Les éditeurs de textes:

VI
Seulement pour ceux qui connaissent cet infâme (mais très complet) éditeur !
Usage: vi toto

MICRO-EMACS
C'est un logiciel du domaine public bien fait et surtout simple d'emploi;
voir la notice fournie avec votre contrat de développeur)
Usage: emacs toto ou: ee toto

ED
Editeur ligne à ligne...
Usage: ed toto

4) Les utilitaires de courrier électronique (email):

NB: Pour simplifier, "nom" correspond au nom de votre correspondant:
exemple nom = francis

MAIL
c'est le plus rustique; à déconseiller.
Usage: mail ou mail nom

MAILX
Plus raffiné que mail, mais inférieur à elm
Usage: mailx ou: mailx nom

ELM
C'est un utilitaire du domaine public, facile d'emploi grâce à sa convivialité
(Attention le mot convivial n'a pas le meme sens sous UNIX que sur l'Amiga !)
Usage: elm ou: elm nom

5) Les interpréteurs interactifs de commandes: les shells

Les shells disponibles sur cbmfra:

sh
C'est le shell par défaut ("Bourne Shell").

csh ("C Shell")
Pour ceux qui aiment jongler avec des commandes parfois non compatibles
avec sh...

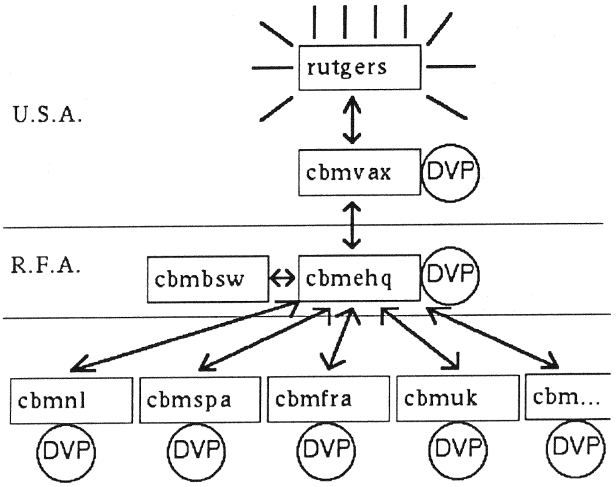
ksh ("Korn Shell")
Totalement compatible avec le Bourne shell; les "aliases" et la commande
"history" sont bien mieux gérés qu' en csh.

Les News

Le serveur est avant tout un lieu de dialogue avec le support de Commodore France grâce au courrier électronique (voir ELM). Mais, il existe aussi la possibilité d'avoir un accès sur la communauté des developpeurs du monde entier et c'est le système des *news* qui permet d'y accéder.

Organisation du réseau

CBMFRA se connecte plusieurs fois par jour sur CBMEHQ (qui est le noeud européen du réseau CBMNET) pour recevoir les informations venant du monde entier.



DVP: développeurs Amiga
cbm...: les serveurs des filiales

Commandes

rn est la commande qui permet l' accès à ces informations (voir le mode d'emploi de *rn*).
Pnews sert exclusivement à écrire dans les news

Organisation des News

L'information est répartie dans des rubriques dont la structure est hiéarchisée; vous avez par exemple le forum *comp.sys.amiga* (c'est à dire computer/system/amiga) où se trouve toute les informations d'ordre général relatives à l'Amiga.
Voici la liste des rubriques ADSP (Amiga Developper Support Program) accessibles par les news:

adsp.hw	Discussions et questions sur le *hardware* de l'Amiga
adsp.sw	Discussions et questions sur le *software* de l'Amiga
adsp.bugs	Rapports de bugs sur l'O.S. de l'Amiga
adsp.docs	Documentations (incluant les *Amiga MAIL*)
adsp.national	Rubrique pour un pays en particulier (ex: adsp.national.french)
adsp.misc	Remplace les rubriques manquantes
adsp.announce	Annonce officielle venant de Commodore
adsp.net	Informations, discussions et questions sur le réseau commodore

Il y a bien sûr des sous-rubriques; par exemple: adsp.net.announce ou adsp.net.questions...

Exemple de procédure de lecture et d'écriture dans les news.

Pour lire les news -> Tapez *rn*
Vous obtenez les messages suivants (variables selon votre configuration):

Unread news in news.group *3 articles*
Unread news in adsp.adsp *8 articles*
etc.
******** *3 unread articles in news.group--read now ? [ynq]*

Vous voulez lire la rubrique suivante ->Tapez *n*
Vous obtenez:

8 Unread articles in adsp.adsp *--read now [ynq]*

Vous voulez lire les messages de cette rubrique -> Tapez *y*

Vous voulez quitter la rubrique avant d'avoir lu tous les messages ->Tapez *q*

Vous voulez lire la rubrique *comp.sources.amig*a -> Tapez *g comp.sources.amiga*

Vous voulez répondre à un message après l'avoir lu -> Tapez *F* (texte cité) ou *f* (texte non cité)

Vous voulez de l'aide -> Tapez *h*

Vous voulez sortir de *rn* -> Tapez *q*

Remarques importantes:

1) Attendez de bien connaître le fonctionnement des news avant d'y écrire quoi que ce soit !
2) Si vous souhaitez poser une question originale ou lancer une nouvelle discussion sans répondre à un message (donc sans utiliser les commandes f ou *F* de *rn*), vous devez impérativement utiliser *Pnews*.

NAME

rn | new read news program

SYNOPSIS

rn [options] [newsgroups]

DESCRIPTION

Rn is a replacement for the readnews(1) program that was written to be as efficient as possible, particularly in human interaction. *Rn* attempts to minimize the amount of "dead" time spent reading news—it tries to get things done while the user is reading or deciding whether to read, and attempts to get useful information onto the screen as soon as possible, highlighting spots that the eye makes frequent reference to, like subjects and previously read lines. Whether or not it's faster, it SEEMS faster.

If no newsgroups are specified, all the newsgroups which have unread news are displayed, and then the user is asked for each one whether he wants to read it, in the order in which the newsgroups occur in the *.newsrc* file. With a list of newsgroups, *rn* will start up in "add" mode, using the list as a set of patterns to add new newsgroups and restrict which newsgroups are displayed. See the discussion of the 'a' command on the newsgroup selection level.

Rn operates on three levels: the newsgroup selection level, the article selection level, and the paging level. Each level has its own set of commands, and its own help menu. At the paging level (the bottom level), *rn* behaves much like the *more*(1) program. At the article selection level, you may specify which article you want next, or read them in the default order, which is either in order of arrival on your system, or by subject threads. At the newsgroup selection level (the top level), you may specify which newsgroup you want next, or read them in the default order, which is the order that the newsgroups occur in your *.newsrc* file. (You will therefore want to rearrange your *.newsrc* file to put the most interesting newsgroups first. This can be done with the 'm' command on the Newsgroup Selection level. WARNING: invoking readnews/vnews (the old user interface) in any way (including as a news checker in your login sequence!) will cause your *.newsrc* to be disarranged again.)

On any level, at ANY prompt, an 'h' may be typed for a list of available commands. This is probably the most important command to remember, so don't you forget it. Typing space to any question means to do the normal thing. You will know what that is because every prompt has a list of several plausible commands enclosed in square brackets. The first command in the list is the one which will be done if you type a space. (All input is done in cbreak mode, so carriage returns should not be typed to terminate anything except certain multilcharacter commands. Those commands will be obvious in the discussion below because they take an argument.)

Upon startup, *rn* will do several things:

1. It will look for your *.newsrc* file, which is your list of subscribedlto newsgroups. If *rn* doesn't find a *.newsrc*, it will create one. If it does find one, it will back it up under the name ".oldnewsrc".
2. It will input your *.newsrc* file, listing out the first several newsgroups with unread news.
3. It will perform certain consistency checks on your *.newsrc*. If your *.newsrc* is out of date in any of several ways, *rn* will warn you and patch it up for you, but you may have to wait a little longer for it to start up.
4. *Rn* will next check to see if any new newsgroups have been created, and give you the opportunity to add them to your *.newsrc*.
5. *Rn* goes into the top prompt level—the newsgroup selection level.

Newsgroup Selection Level

In this section the words "next" and "previous" refer to the ordering of the newsgroups in your *.newsrc* file. On the newsgroup selection level, the prompt looks like this:

******** 17 unread articles in net.blurfl—read now? [ynq]

and the following commands may be given at this level:

y,SP Do this newsgroup now.

.command
Do this newsgroup now, but execute *command* before displaying anything. The command will be interpreted as if given on the article selection level.

= Do this newsgroup now, but list subjects before displaying articles.

n Go to the next newsgroup with unread news.

N Go to the next newsgroup.

p Go to the previous newsgroup with unread news. If there is none, stay at the current newsgroup.

P Go to the previous newsgroup.

– Go to the previously displayed newsgroup (regardless of whether it is before or after the current one in the list).

1 Go to the first newsgroup.

^ Go to the first newsgroup with unread news.

$ Go to the end of the newsgroups list.

g newsgroup
Go to *newsgroup*. If it isn't currently subscribed to, you will be asked if you want to subscribe.

/pattern Scan forward for a newsgroup matching *pattern*. Patterns do globbing like filenames, i.e., use ? to match a single character, * to match any sequence of characters, and [] to specify a list of characters to match. ("all" may be used as a synonym for "*".) Unlike normal filename globbing, newsgroup searching is not anchored to the front and back of the filename, i.e. "/jok" will find net.jokes. You may use ^ or $ to anchor the front or back of the search: "/^test$" will find newsgroup test and nothing else If you want to include newsgroups with 0 unread articles, append /r. If the newsgroup is not found between the current newsgroup and the last newsgroup, the search will wrap around to the beginning.

?pattern Same as /, but search backwards.

u Unsubscribe from current newsgroup.

l string List newsgroups not subscribed to which contain the string specified.

L Lists the current state of the *.newsrc*, along with status information.

Status	Meaning
<number>	Count of unread articles in newsgroup.
READ	No unread articles in newsgroup.
UNSUB	Unsubscribed newsgroup.
BOGUS	Bogus newsgroup.
JUNK	Ignored line in .newsrc (e.g. readnews "options" line).

(A bogus newsgroup is one that is not in the list of active newsgroups in the active file, which on most systems is /usr/lib/news/active.)

m name Move the named newsgroup somewhere else in the *.newsrc*. If no name is given, the current newsgroup is moved. There are a number of ways to specify where you want the newsgroup—type h for help when it asks where you want to put it.

c Catch up—mark all unread articles in this newsgroup as read.

o pattern Only display those newsgroups whose name matches *pattern*. Patterns are the same as for the '/' command. Multiple patterns may be separated by spaces, just as on the command line. The restriction will remain in effect either until there are no articles left in the restricted set of newsgroups, or another restriction command is given. Since *pattern* is optional, 'o' by itself will remove the restriction.

a pattern Add new newsgroups matching *pattern*. Newsgroups which are already in your *.newsrc* file, whether subscribed to or not, will not be listed. If any new newsgroups are found, you will be asked for each one whether you would like to add it. After any new newsgroups have been added, the 'a' command also restricts the current set of newsgroups just like the 'o' command does.

& Print out the current status of command line switches and any newsgroup restrictions.

&switch {switch}
Set additional command line switches.

&& Print out the current macro definitions.

&&keys commands
Define additional macros.

!command
Escape to a subshell. One exclamation mark (!) leaves you in your own news directory. A double exclamation mark (!!) leaves you in the spool directory for news, which on most systems is /usr/spool/news. The environment variable SHELL will be used if defined. If *command* is null, an interactive shell is started.

q Quit.

x Quit, restoring .newsrc to its state at startup of *rn*. The .newsrc you would have had if you had exited with 'q' will be called .newnewsrc, in case you didn't really want to type 'x'.

^K Edit the global KILL file. This is a file which contains /pattern/j commands (one per line) to be applied to every newsgroup as it is started up, that is, when it is selected on the newsgroup selection level. The purpose of a KILL file is to mark articles as read on the basis of some set of patterns. This saves considerable wear and tear on your 'n' key. There is also a local KILL file for each newsgroup. Because of the overhead involved in searching for articles to kill, it is better if possible to use a local KILL file. Local KILL files are edited with a '^K' on the article selection level. There are also automatic ways of adding search commands to the local KILL file—see the 'K' command and the K search modifier on the article selection level.

If either of the environment variables VISUAL or EDITOR is set, the specified editor will be invoked; otherwise a default editor (normally vi(1)) is invoked on the KILL file.

Article Selection Level

On the article selection level, *rn* selects (by default) unread articles in numerical order (the order in which articles have arrived at your site). If you do a subject search (^N), the default order is modified to be numerical order within each subject thread. You may switch back and forth between numerical order and subject thread order at will. The –S switch can be used to make subject search mode the default.

On the article selection level you are *not* asked whether you want to read an article before the article is displayed; rather, *rn* simply displays the first page (or portion of a page, at low baud rates) of the article and asks if you want to continue. The normal article selection prompt comes at the END of the article (though article selection commands can be given from within the middle of the article (the pager level) also). The prompt at the end of an article looks like this:

End of article 248 (of 257)—what next? [npq]

The following are the options at this point:

n,SP Scan forward for next unread article. (Note: the 'n' (next) command when typed at the end of an article does not mark the article as read, since an article is automaticaly marked as read after the last line of it is printed. It is therefore possible to type a sequence such as 'mn' and leave the article marked as unread. The fact that an article is marked as read by typing 'n', 'N', '^N', 's', or 'S' within the MIDDLE of the article is in fact a special case.)

N Go to the next article.

^N Scan forward for the next article with the same subject, and make ^N default (subject search mode).

p Scan backward for previous unread article. If there is none, stay at the current article.

P Go to the previous article.

– Go to the previously displayed article (regardless of whether that article is before or after this article in the normal sequence).

^P Scan backward for the previous article with the same subject, and make ^N default (subject search mode).

^R Restart the current article.

v Restart the current article verbosely, displaying the entire header.

^L Refresh the screen.

^X Restart the current article, and decrypt as a rot13 message.

X Refresh the screen, and decrypt as a rot13 message.

b Back up one page.

q Quit this newsgroup and go back to the newsgroup selection level.

^ Go to the first unread article.

$ Go to the last article (actually, one past the last article).

number Go to the numbered article.

range{,range} command{:command}
Apply a set of commands to a set of articles. A range consists of either <article number> or <article number>–<article number>. A dot '.' represents the current article, and a dollar sign '$' represents the last article.

Applicable commands include 'm' (mark as unread), 'M' (delayed mark as unread), 'j' (mark as read), "s dest" (save to a destination), "!command" (shell escape), "=" (print the subject) and "C" (cancel).

j Junk the current article—mark it as read. If this command is used from within an article, you are left at the end of the article, unlike 'n', which looks for the next article.

m Mark the current article as still unread. (If you are in subject search mode you probably want to use M instead of m. Otherwise the current article may be selected as the beginning of the next subject thread.)

M Mark the current article as still unread, but not until the newsgroup is exited. Until then, the current article will be marked as read. This is useful for returning to an article in another session, or in another newsgroup.

/pattern Scan forward for article containing *pattern* in the subject. See the section on Regular Expressions. Together with the escape substitution facility described later, it becomes easy to search for various attributes of the current article, such as subject, article ID, author name, etc. The previous

pattern can be recalled with "<esc>/". If *pattern* is omitted, the previous pattern is assumed.

/pattern/h
Scan forward for article containing *pattern* in the header.

/pattern/a
Scan forward for article containing *pattern* anywhere in article.

/pattern/r Scan read articles also.

/pattern/c
Make search case sensitive. Ordinarily upper and lower case are considered the same.

/pattern/modifiers:command{:command}
Apply the commands listed to articles matching the search command (possibly with h, a, or r modifiers). Applicable commands include 'm' (mark as unread), 'M' (delayed mark as unread), 'j' (mark as read), "s dest" (save to a destination), "!command" (shell escape), "=" (print the subject) and "C" (cancel). If the first command is 'm' or 'M', modifier r is assumed. A K may be included in the modifiers (not the commands) to cause the entire command (sans K) to be saved to the local KILL file, where it will be applied to every article that shows up in the newsgroup.

For example, to save all articles in a given newsgroup to the line printer and mark them read, use "/^/|lpr:j". If you say "/^/K|lpr:j", this will happen every time you enter the newsgroup.

?pattern Scan backward for article containing *pattern* in the subject. May be modified as the forward search is: ?pattern?modifiers[:commands]. It is likely that you will want an r modifier when scanning backward.

k Mark as read all articles with the same subject as the current article. (Note: there is no single character command to temporarily mark as read (M command) articles matching the current subject. That can be done with "/<esc>s/M", however.)

K Do the same as the k command, but also add a line to the local KILL file for this newsgroup to kill this subject every time the newsgroup is started up. For a discussion of KILL files, see the "^K' command below. See also the K modifier on searches above.

^K Edit the local KILL file for this newsgroup. Each line of the KILL file should be a command of the form /pattern/j. (With the exception that *rn* will insert a line at the beginning of the form "THRU <number>", which tells *rn* the maximum article number that the KILL file has been applied to. You may delete the THRU line to force a rescan of current articles.) You may also have reason to use the m, h, or a modifiers. Be careful with the M modifier in a kill file—there are more efficient ways to never read an article. You might have reason to use it if a particular series of articles is posted to multiple newsgroups. In this case, M would force you to view the article in a different newsgroup.

To see only newgroup articles in the control newsgroup, for instance, you might put

/^/j
/newgroup/m

which kills all subjects not containing "newgroup". You can add lines automatically via the K command and K search modifiers, but editing is the only way to remove lines. If either of the environment variables VISUAL or EDITOR is set, the specified editor will be invoked; otherwise a default editor (normally vi) is invoked on the KILL file.

The KILL file may also contain switch setting lines beginning with '&'. Additionally, any line beginning with 'X' is executed on exit from the newsgroup rather than on entrance. This can be used to set switches back to a default value.

r Reply through net mail. The environment variables MAILPOSTER and MAILHEADER may be used to modify the mailing behavior of *rn* (see environment section). If on a nonexistent article such as the "End of newsgroup" pseudolarticle (which you can get to with a '$' command), invokes the mailer to nobody in particular.

R Reply, including the current article in the header file generated. (See 'F' command below). The YOUSAID environment variable controls the format of the attribution line.

f Submit a followup article. If on a nonexistent article such as the "End of newsgroup" pseudol article (which you can get to with a '$' command), posts an original article (basenote).

F Submit a followup article, and include the old article, with lines prefixed either by ">" or by the argument to a **–F** switch. *Rn* will attempt to provide an attribution line in front of the quoted article, generated from the From: line of the article. Unfortunately, the From: line doesn't always contain the right name; you should double check it against the signature and change it if necessary, or you may have to apologize for quoting the wrong person. The environment variables NEWSPOSTER, NEWSHEADER and ATTRIBUTION may be used to modify the posting behavior of *rn* (see environment section).

C Cancel the current article, but only if you are the contributor or superuser.

c Catch up in this newsgroup; i.e., mark all articles as read.

u Unsubscribe to this newsgroup.

s destination

Save to a filename or pipe using sh. If the first character of the destination is a vertical bar, the rest of the command is considered a shell command to which the article is passed through standard input. The command is subject to filename expansion. (See also the environment variable PIPESAVER.) If the destination does not begin with a vertical bar, the rest of the command is assumed to be a filename of some sort. An initial tilde "~" will be translated to the name of the home directory, and an initial environment variable substitution is also allowed. If only a directory name is specified, the environment variable SAVENAME is used to generate the actual name. If only a filename is specified (i.e. no directory), the environment variable SAVEDIR will be used to generate the actual directory. If nothing is specified, then obviously both variables will be used. Since the current directory for rn while doing a save command is your private news directory, saying "s ./filename" will force the file to your news directory. Save commands are also run through % interpretation, so that you can say "s %O/filename" to save to the directory you were in when you ran *rn*, and "s %t" to save to a filename consisting of the Internet address of the sender.

After generating the full pathname of the file to save to, *rn* determines if the file exists already, and if so, appends to it. *Rn* will attempt to determine if an existing file is a mailbox or a normal file, and save the article in the same format. If the output file does not yet exist, *rn* will by default ask you which format you want, or you can make it skip the question with either the **–M** or **–N** switch. If the article is to be saved in mailbox format, the command to do so is generated from the environment variable MBOXSAVER. Otherwise, NORMSAVER is used.

S destination

Save to a filename or pipe using a preferred shell, such as csh. Which shell is used depends first on what you have the environment variable SHELL set to, and in the absence of that, on what your news administrator set for the preferred shell when he or she installed *rn*.

| command

Shorthand for "s | command".

w destination
The same as "s destination", but saves without the header.

W destination
The same as "S destination", but saves without the header.

& Print out the current status of command line switches.

&switch {switch}
Set additional command line switches.

&& Print out current macro definitions.

&&keys commands
Define an additional macro.

!command
Escape to a subshell. One exclamation mark (!) leaves you in your own news directory. A double exclamation mark (!!) leaves you in the spool directory of the current newsgroup. The environment variable SHELL will be used if defined. If *command* is null, an interactive shell is started.

You can use escape key substitutions described later to get to many runltime values. The command is also run through % interpretation, in case it is being called from a range or search command.

= List subjects of unread articles.

Print last article number.

Pager Level

At the pager level (within an article), the prompt looks like this:

—MORE—(17%)

and a number of commands may be given:

SP Display next page.

x Display next page and decrypt as a rot13 message.

d,^D Display half a page more.

CR Display one more line.

q Go to the end of the current article (don't mark it either read or unread). Leaves you at the "What next?" prompt.

j Junk the current article. Mark it read and go to the end of the article.

^L Refresh the screen.

X Refresh the screen and decrypt as a rot13 message.

b,^B Back up one page.

gpattern Goto (search forward for) *pattern* within current article. Note that there is no space between the command and the pattern. If the pattern is found, the page containing the pattern will be displayed. Where on the page the line matching the pattern goes depends on the value of the –g switch. By default the matched line goes at the top of the screen.

G Search for g pattern again.

^G This is a special version of the 'g' command that is for skipping articles in a digest. It is equivalent to setting "–g4" and then executing the command "g^Subject:".

TAB This is another special version of the 'g' command that is for skipping inclusions of older articles. It is equivalent to setting "–g4" and then executing the command "g^[^c]", where *c* is the first character of the last line on the screen. It searches for the first line that doesn't begin with the same character as the last line on the screen.

!command
Escape to a subshell.

The following commands skip the rest of the current article, then behave just as if typed to the "What next?" prompt at the end of the article. See the documentation at the article selection level for these commands.

$ & / = ? c C f F k K ^K m M r R ^R u v Y ^
number
range{,range} command{:command}

The following commands also skip to the end of the article, but have the additional effect of marking the current article as read:

n N ^N s S | w W

Miscellaneous facts about commands

An 'n' typed at either the "Last newsgroup" prompt or a "Last article" prompt will cycle back to the top of the newsgroup or article list, whereas a 'q' will quit the level. (Note that 'n' does not mean "no", but rather "next".) A space will of course do whatever is shown as the default, which will vary depending on whether rn thinks you have more articles or newsgroups to read.

The 'b' (backup page) command may be repeated until the beginning of the article is reached. If *rn* is suspended (via a ^Z), then when the job is resumed, a refresh (^L) will automatically be done (Berkeleyltype systems only). If you type a command such as '!' or 's' which takes you from the middle of the article to the end, you can always get back into the middle by typing "^L'.

In multilcharacter commands such as '!', 's', '/', etc, you can interpolate various runltime values by typing escape and a character. To find out what you can interpolate, type escape and 'h', or check out the single character % substitutions for environment variables in the Interpretation and Interpolation section, which are the same. Additionally, typing a double escape will cause any % substitutions in the string already typed in to be expanded.

Options

Rn has a nice set of options to allow you to tailor the interaction to your liking. (You might like to know that the author swears by "–e –m –S –/".) These options may be set on the command line, via the RNINIT environment variable, via a file pointed to by the RNINIT variable, or from within rn via the & command. Options may generally be unset by saying "+switch". Options include:

–c checks for news without reading news. If a list of newsgroups is given on the command line, only those newsgroups will be checked; otherwise all subscribedlto newsgroups are checked. Whenever the **–c** switch is specified, a nonlzero exit status from *rn* means that there is unread news in one of the checked newsgroups. The **–c** switch does not disable the printing of newsgroups with unread news; this is controlled by the **–s** switch. (The **–c** switch is not meaningful when given via the & command.)

–C<number>
tells *rn* how often to checkpoint the *.newsrc*, in articles read. Actually, this number says when to start thinking about doing a checkpoint if the situation is right. If a reasonable checkpointing situation doesn't arise within 10 more articles, the *.newsrc* is checkpointed willylnilly.

–d<directory name>
sets the default save directory to something other than ~/News. The directory name will be globbed

(via csh) if necessary (and if possible). Articles saved by *rn* may be placed in the save directory or in a subdirectory thereof depending on the command that you give and the state of the environment variables SAVEDIR and SAVENAME. Any KILL files (see the K command in the Article Selection section) also reside in this directory and its subdirectories, by default. In addition, shell escapes leave you in this directory.

–D<flags>
enables debugging output. See common.h for flag values. Warning: normally *rn* attempts to restore your *.newsrc* when an unexpected signal or internal error occurs. This is disabled when any debugging flags are set.

–e causes each page within an article to be started at the top of the screen, not just the first page. (It is similar to the **–c** switch of *more*(1).) You never have to read scrolling text with this switch. This is helpful especially at certain baud rates because you can start reading the top of the next page without waiting for the whole page to be printed. It works nicely in conjuction with the **–m** switch, especially if you use halflintensity for your highlight mode. See also the **–L** switch.

–E<name>=<val>
sets the environment variable <name> to the value specified. Within *rn*, "&–ESAVENAME=%t" is similar to "setenv SAVENAME '%t'" in *csh*, or "SAVENAME='%t'; export SAVENAME" in *sh*. Any environment variables set with **–E** will be inherited by subprocesses of *rn*.

–F<string>
sets the prefix string for the 'F' followup command to use in prefixing each line of the quoted article. For example, "–F<tab>" inserts a tab on the front of each line (which will cause long lines to wrap around, unfortunately), "–F>>>>" inserts ">>>>" on every line, and "–F" by itself causes nothing to be inserted, in case you want to reformat the text, for instance. The initial default prefix is ">".

–g<line>
tells *rn* which line of the screen you want searchedlfor strings to show up on when you search with the 'g' command within an article. The lines are numbered starting with 1. The initial default is "–g1", meaning the first line of the screen. Setting the line to less than 1 or more than the number of lines on the screen will set it to the last line of the screen.

–h<string>
hides (disables the printing of) all header lines beginning with *string*. For instance, –hexp will disable the printing of the "Expires:" line. Case is insignificant. If <string> is null, all header lines except Subject are hidden, and you may then use **+h** to select those lines you want to see. You may wish to use the baudlrate switch modifier below to hide more lines at lower baud rates.

–H<string>
works just like **–h** except that instead of setting the hiding flag for a header line, it sets the magic flag for that header line. Certain header lines have magic behavior that can be controlled this way. At present, the following actions are caused by the flag for the particular line: the Newsgroups line will only print when there are multiple newsgroups, the Subject line will be underlined, and the Expires line will always be suppressed if there is nothing on it. In fact, all of these actions are the default, and you must use **+H** to undo them.

–i=<number>
specifies how long (in lines) to consider the initial page of an article—normally this is determined automatically depending on baud rate. (Note that an entire article header will always be printed regardless of the specified initial page length. If you are working at low baud rate and wish to reduce the size of the headers, you may hide certain header lines with the **h** switch.)

–l disables the clearing of the screen at the beginning of each article, in case you have a bizarre terminal.

–L tells *rn* to leave information on the screen as long as possible by not blanking the screen between pages, and by using clear to end|of|line. (The *more*(1) program does this.) This feature works only if you have the requisite termcap capabilities. The switch has no effect unless the **–e** switch is set.

–m=<mode>
enables the marking of the last line of the previous page printed, to help the user see where to continue reading. This is most helpful when less than a full page is going to be displayed. It may also be used in conjunction with the **–e** switch, in which case the page is erased, and the first line (which is the last line of the previous page) is highlighted. If **–m=s** is specified, the standout mode will be used, but if **–m=u** is specified, underlining will be used. If neither =s or =u is specified, standout is the default. Use **+m** to disable highlighting.

–M forces mailbox format in creating new save files. Ordinarily you are asked which format you want.

–N forces normal (non|mailbox) format in creating new save files. Ordinarily you are asked which format you want.

–r causes *rn* to restart in the last newsgroup read during a previous session with *rn*. It is equivalent to starting up normally and then getting to the newsgroup with a g command.

–s with no argument suppresses the initial listing of newsgroups with unread news, whether **–c** is specified or not. Thus **–c** and **–s** can be used together to test "silently" the status of news from within your *.login* file. If **–s** is followed by a number, the initial listing is suppressed after that many lines have been listed. Presuming that you have your *.newsrc* sorted into order of interest, **–s5** will tell you the 5 most interesting newsgroups that have unread news. This is also a nice feature to use in your *.login* file, since it not only tells you whether there is unread news, but also how important the unread news is, without having to wade through the entire list of unread newsgroups. If no **–s** switch is given **–s5** is assumed, so just putting "rn –c" into your .login file is fine.

–S<number>
causes *rn* to enter subject search mode (^N) automatically whenever a newsgroup is started up with <number> unread articles or more. Additionally, it causes any 'n' typed while in subject search mode to be interpreted as '^N' instead. (To get back out of subject search mode, the best command is probably '^'.) If <number> is omitted, 3 is assumed.

–t puts *rn* into terse mode. This is more cryptic but useful for low baud rates. (Note that your system administrator may have compiled *rn* with either verbose or terse messages only to save memory.) You may wish to use the baud|rate switch modifier below to enable terse mode only at lower baud rates.

–T allows you to type ahead of rn. Ordinarily rn will eat typeahead to prevent your autorepeating space bar from doing a very frustrating thing when you accidentally hold it down. If you don't have a repeating space bar, or you are working at low baud rate, you can set this switch to prevent this behavior. You may wish to use the baud|rate switch modifier below to disable typeahead only at lower baud rates.

–v sets verification mode for commands. When set, the command being executed is displayed to give some feedback that the key has actually been typed. Useful when the system is heavily loaded and you give a command that takes a while to start up.

–/ sets SAVEDIR to "%p/%c" and SAVENAME to "%a", which means that by default articles are saved in a subdirectory of your private news directory corresponding to the name of the the current newsgroup, with the filename being the article number. +/ sets SAVEDIR to "%p" and SAVENAME to "%^C", which by default saves articles directly to your private news directory, with the filename being the name of the current newsgroup, first letter capitalized. (Either +/ or –/ may be default on your system, depending on the feelings of your news administrator when he, she or it installed *rn*.) You may, of course, explicitly set SAVEDIR and SAVENAME to other values—see discussion in the environment section.

Any switch may be selectively applied according to the current baud|rate. Simply prefix the switch with +speed to apply the switch at that speed or greater, and –speed to apply the switch at that speed or less. Examples: –1200–hposted suppresses the Posted line at 1200 baud or less; +9600–m enables marking at 9600 baud or more. You can apply the modifier recursively to itself also: +300–1200–t sets terse mode from 300 to 1200 baud.

Similarly, switches may be selected based on terminal type:

```
–=vt100+T                 set +T on vt100
–=tvi920–ETERM=mytvi      get a special termcap entry
–=tvi920–ERNMACRO=%./.rnmac.tvi
                          set up special keymappings
+=paper–v                 set verify mode if not hardcopy
```

Some switch arguments, such as environment variable values, may require spaces in them. Such spaces should be quoted via ", ', or \ in the conventional fashion, even when passed via RNINIT or the & command.

Regular Expressions

The patterns used in article searching are regular expressions such as those used by *ed*(1). In addition, \w matches an alphanumeric character and \W a nonalphanumeric. Word boundaries may be matched by \b, and non|boundaries by \B. The bracketing construct \(... \) may also be used, and \digit matches the digit'th substring, where digit can range from 1 to 9. \0 matches whatever the last bracket match matched. Up to 10 alternatives may given in a pattern, separated by \|, with the caveat that \(... \| ... \) is illegal.

Interpretation and Interpolation

Many of the strings that *rn* handles are subject to interpretations of several types. Under filename expansion, an initial "~/" is translated to the name of your home directory, and "~name" is translated to the login directory for the user specified. Filename expansion will also expand an initial environment variable, and also does the backslash, uparrow and percent expansion mentioned below.

All interpreted strings go through backslash, uparrow and percent interpretation. The backslash escapes are the normal ones (such as \n, \t, \nnn, etc.). The uparrow escapes indicate control codes in the normal fashion. Backslashes or uparrows to be passed through should be escaped with backslash. The special percent escapes are similar to printf percent escapes. These cause the substitution of various run|time values into the string. The following are currently recognized:

%a Current article number.

%A Full name of current article (%P/%c/%a). (On a Eunice system with the LINKART option, %P/%c/%a returns the name of the article in the current newsgroup, while %A returns the real name of the article, which may be different if the current article was posted to multiple newsgroups.)

%b Destination of last save command, often a mailbox.

%B The byte offset to the beginning of the part of the article to be saved, set by the save command. The 's' and 'S' commands set it to 0, and the 'w' and 'W' commands set it to the byte offset of the body of the article.

%c Current newsgroup, directory form.

%C Current newsgroup, dot form.

%d Full name of newsgroup directory (%P/%c).

%D "Distribution:" line from the current article.

%f "From:" line from the current article, or the "Reply|To:" line if there is one. This differs from %t in that comments (such as the full name) are not stripped out with %f.

%F "Newsgroups:" line for a new article, constructed from "Newsgroups:" and "Followup|To:" lines of current article.

%h Name of the header file to pass to the mail or news poster, containing all the information that the poster program needs in the form of a message header. It may also contain a copy of the current article. The format of the header file is controlled by the MAILHEADER and NEWSHEADER environment variables.

%H Host name (your machine's name).

%i "Message|I.D.:" line from the current article, with <> guaranteed.

%I The reference indication mark (see the **-F** switch.)

%l The news administrator's login name, if any.

%L Login name (yours).

%m The current mode of rn, for use in conditional macros.

i	Initializing.
n	Newsgroup selection level.
a	Article selection level (What next?).
p	Pager level (MORE prompt).
A	Add this newsgroup?
C	Catchup confirmation.
D	Delete bogus newsgroups?
M	Use mailbox format?
R	Resubscribe to this newsgroup?

Note that yes/no questions are all upper|case modes. If, for example, you wanted to disallow defaults on all yes/no questions, you could define the following macro:

\040 %(%m=[A|Z]?h:)

%M The number of articles marked to return via the 'M' command. If the same article is Marked multiple times, "%M" counts it multiple times in the current implementation.

%n "Newsgroups:" line from the current article.

%N Full name (yours).

%o Organization (yours).

%O Original working directory (where you ran rn from).

%p Your private news directory, normally ~/News.

%P Public news spool directory, normally /usr/spool/news.

%r Last reference on references line of current article (parent article id).

%R References list for a new article, constructed from the references and article ID of the current article.

%s Subject, with all Re's and (nf)'s stripped off.

%S Subject, with one "Re:" stripped off.

%t "To:" line derived from the "From:" and "Reply|To:" lines of the current article. This always returns an Internet format address.

%T "To:" line derived from the "Path:" line of the current article to produce a uucp path.

%u The number of unread articles in the current newsgroup.

%U The number of unread articles in the current newsgroup, not counting the current article.

%x The news library directory.

%X The rn library directory.

%z The length of the current article in bytes.

%~ Your home directory.

%. The directory containing your dot files, which is your home directory unless the environment variable DOTDIR is defined when rn is invoked.

%$ Current process number.

%/ Last search string.

%% A percent sign.

%{name} or %{name–default}
The environment variable "name".

%[name] The value of header line "Name:" from the current article. The "Name: " is not included. For example "%D" and "%[distribution]" are equivalent. The name must be spelled out in full.

%`command`
Inserts the output of the command, with any embedded newlines translated to space.

%"prompt"
Prints prompt on the terminal, then inputs one string, and inserts it.

%(test_text=pattern?then_text:else_text)
If *test_text* matches *pattern*, has the value *then_text*, otherwise *else_text*. The ":else_text" is optional, and if absent, interpolates the null string. The = may be replaced with != to negate the test. To quote any of the metacharacters ('=', '?', ':', or ')'), precede with a backslash.

%digit The digits 1 through 9 interpolate the string matched by the nth bracket in the last pattern match that had brackets. If the last pattern had alternatives, you may not know the number of the bracket you want— %0 will give you the last bracket matched.

Modifiers: to capitalize the first letter, insert '^': "%^C" produces something like "Net.jokes". Inserting '_' causes the first letter following the last '/' to be capitalized: "%_c" produces "net/Jokes".

ENVIRONMENT

The following environment variables are paid attention to by *rn*. In general the default values assumed for these variables by *rn* are reasonable, so if you are using *rn* for the first time, you can safely ignore this section. Note that the defaults below may not correspond precisely to the defaults on your system. To find the actual defaults you would need to look in config.h and common.h in the rn source directory, and the file INIT in the rn library.

Those variables marked (%) are subject to % interpolation, and those marked (~) are subject to both % interpolation and ~ interpretation.

ATTRIBUTION (%)
Gives the format of the attribution line in front of the quoted article included by an F command.

Default: In article %i %f writes:

CANCEL (~)
The shell command used to cancel an article.

Default: inews –h < %h

CANCELHEADER (%)
The format of the file to pass to the CANCEL command in order to cancel an article.

Default:
Newsgroups: %n
Subject: cmsg cancel %i
References: %R
Reply|To: %L@%H.UUCP (%N)
Distribution: %D
Organization: %o

%i cancelled from rn.

DOTDIR
Where to find your dot files, if they aren't in your home directory. Can be interpolated using "%.".

Default: $HOME

EDITOR (~)
The name of your editor, if VISUAL is undefined.

Default: whatever your news administrator compiled in, usually vi.

FIRSTLINE (%)
Controls the format of the line displayed at the top of an article. Warning: this may go away.

Default: Article %a %(%U%M!=^00$?(%U more%(%M!=^0$? + %M Marked to return)\))in %C:, more or less.

HIDELINE
If defined, contains a regular expression which matches article lines to be hidden, in order, for instance, to suppress quoted material. A recommended string for this purpose is "^>...", which *doesn't* hide lines with only '>', to give some indication that quoted material is being skipped. If you want to hide more than one pattern, you can use "|" to separate the alternatives. You can view the hidden lines by restarting the article with the 'v' command.

There is some overhead involved in matching each line of the article against a regular expression. You might wish to use a baud|rate modifier to enable this feature only at low baud rates.

Default: undefined

HOME Your home directory. Affects ~ interpretation, and the location of your dot files if DOTDIR is not defined.

Default: $LOGDIR

KILLGLOBAL (~)
Where to find the KILL file to apply to every newsgroup. See the '^K' command at the newsgroup selection level.

Default: %p/KILL

KILLLOCAL (~)
Where to find the KILL file for the current newsgroup. See the commands 'K' and '^K' at the article selection level, and the search modifier 'K'.

Default: %p/%c/KILL

LOGDIR
Your home directory if HOME is undefined. Affects ~ interpretation, and the location of your dot files if DOTDIR is not defined.

Default: none.

Explanation: you must have either $HOME or $LOGDIR.

LOGNAME
Your login name, if USER is undefined. May be interpolated using ''%L''.

Default: value of getlogin().

MAILCALL (~)
What to say when there is new mail.

Default: (Mail)

MAILFILE (~)
Where to check for mail.

Default: /usr/spool/mail/%L

MAILHEADER (%)
The format of the header file for replies. See also MAILPOSTER.

Default:

To: %T
Subject: %(%i=^$?:Re: %S
Newsgroups: %n
In\Reply\To: %i)
%(%[references]!=^$?References\: %[references]
)Organization: %o
Cc:
Bcc: \n\n

MAILPOSTER (~)
The shell command to be used by the reply commands (r and R) in order to allow you to enter and deliver the response. *Rn* will not itself call upon an editor for replies—this is a function of the program called by *rn*. See also MAILHEADER.

Default: Rnmail –h %h

MBOXSAVER (~)
The shell command to save an article in mailbox format.

Default: %X/mbox.saver %A %P %c %a %B %C "%b" \
"From: %T %'date'"

Explanation: the first seven arguments are the same as for NORMSAVER. The eighth argument to the shell script is the new From: line for the article, including the posting date, derived either directly from the Posted: line, or notlsoldirectly from the Date: line. Header munging at its finest.

NAME Your full name. May be interpolated using ''%N''.

Default: name from /etc/passwd, or ~/.fullname.

NEWSHEADER (%)
The format of the header file for followups. See also NEWSPOSTER.

Default:

Newsgroups: %(%F=^$?%C:%F)

Subject: %(%S=^$?%"0ubject: ":Re: %S)
Summary:
Expires:
%(%R=^$?:References: %R
)Sender:
Reply|To: %L@%H.UUCP (%N)
Follow|up|To:
Distribution: %(%i=^$?%"0istribution: ":%D)
Organization: %o
Keywords: \n\n

NEWSPOSTER (~)
The shell command to be used by the followup commands (f and F) in order to allow you to enter and post a followup news article. *Rn* will not itself call upon an editor for followups—this is a function of the program called by *rn*. See also NEWSHEADER.

Default: Pnews –h %h

NORMSAVER (~)
The shell command to save an article in the normal (non|mailbox) format.

Default: %X/norm.saver %A %P %c %a %B %C "%b"

ORGANIZATION
Either the name of your organization, or the name of a file containing the name of your organization. May be interpolated using ''%o''.

Default: whatever your news administrator compiled in.

PAGESTOP
If defined, contains a regular expression which matches article lines to be treated as form|feeds. There are at least two things you might want to do with this. To cause page breaks between articles in a digest, you might define it as ''^||||||||''. To force a page break before a signature, you could define it as ''^|| $''. (Then, when you see ''||'' at the bottom of the page, you can skip the signature if you so desire by typing 'n' instead of space.) To do both, you could use ''^||''. If you want to break on more than one pattern, you can use '' | '' to separate the alternatives.

There is some overhead involved in matching each line of the article against a regular expression. You might wish to use a baud|rate modifier to enable this feature only at low baud rates.

Default: undefined

PIPESAVER (%)
The shell command to execute in order to accomplish a save to a pipe (''s | command'' or ''w | command''). The command typed by the user is substituted in as %b.

Default: %(%B=^0$?<%A:tail +%Bc %A |) %b

Explanation: if %B is 0, the command is ''<%A %b'', otherwise the command is ''tail +%Bc %A | %b''.

RNINIT Default values for switches may be passed to *rn* by placing them in RNINIT. Any switch that is set in RNINIT may be overruled on the command line, or via the '&' command from within *rn*. Binary|valued switches that are set with ''–switch'' may be unset using ''+switch''.

If RNINIT begins with a '/' it is assumed to be the name of a file containing switches. If you want to set many environment variables but don't want to keep them all in your environment, or if the use of any of these variables conflicts with other programs, you can use this feature along with the –E switch to set the environment variables upon startup.

Default: '' ''.

RNMACRO (~)
The name of the file containing macros and key mappings. See the MACROS section.

Default: %./.rnmac

SAVEDIR (~)
The name of the directory to save to, if the save command does not specify a directory name.

Default:
If –/ is set: %p/%c
If +/ is set: %p

SAVENAME (%)
The name of the file to save to, if the save command contains only a directory name.

Default:
If –/ is set: %a
If +/ is set: %^C

SHELL The name of your preferred shell. It will be used by the '!', 'S' and 'W' commands.

Default: whatever your news administrator compiled in.

SUBJLINE (%)
Controls the format of the lines displayed by the '=' command at the article selection level.

Default: %s

TERM Determines which termcap entry to use, unless TERMCAP contains the entry.

TERMCAP
Holds either the name of your termcap file, or a termcap entry.

Default: /etc/termcap, normally.

USER Your login name. May be interpolated using "%L".

Default: $LOGNAME

VISUAL (~)
The name of your editor.

Default: $EDITOR

YOUSAID (%)
Gives the format of the attribution line in front of the quoted article included by an R command.

Default: In article %i you write:

MACROS

When *rn* starts up, it looks for a file containing macro definitions (see environment variable RNMACRO). Any sequence of commands may be bound to any sequence of keys, so you could remap your entire keyboard if you desire. Blank lines or lines beginning with # in the macro file are considered comments; otherwise *rn* looks for two fields separated by white space. The first field gives the sequence of keystrokes that trigger the macro, and the second field gives the sequence of commands to execute. Both fields are subject to % interpolation, which will also translate backslash and uparrow sequences. (The keystroke field is interpreted at startup time, but the command field is interpreted at macro execution time so that you may refer to % values in a macro.) For example, if you want to reverse the roles of carriage return and space in rn

```
^J      \040
^M      \040
\040    ^J
```

will do just that. By default, all characters in the command field are interpreted as the canonical *rn* characters, i.e. no macro expansion is done. Otherwise the above pair of macros would cause an infinite loop. To force macro expansion in the command field, enclose the macro call with ^(... ^) thusly:

```
@s      |mysavescript
@w      w^(@s^)
```

You can use the %() conditional construct to construct macros that work differently under different circumstances. In particular, the current mode (%m) of *rn* could be used to make a command that only works at a particular level. For example,

```
^[[O    %(%m=p?\040)
```

will only allow the macro to work at the pager level.

```
%(%{TERM}=vt100?^[[O        /^J
```

will do the binding only if the terminal type is vt100, though if you have many of these it would be better to have separate files for each terminal.

If you want to bind a macro to a function key that puts a common garbage character after the sequence (such as the carriage return on the end of Televideo 920 function sequences), DO NOT put the carriage return into all the sequences or you will waste a CONSIDERABLE amount of internal storage. Instead of "^AF^M", put "^AF+1", which indicates to *rn* that it should gobble up one character after the F.

AUTHOR

Larry Wall <lwall@sdcrdcf.UUCP>
Regular expression routines are borrowed from emacs, by James Gosling.

FILES

%./.newsrc	status of your news reading
%./.oldnewsrc	backup copy of your *.newsrc* from start of session
%./.rnlock	lock file so you don't screw up your *.newsrc*
%./.rnlast	info from last run of rn
%./.rnsoft	soft pointers into /usr/lib/active to speed startup, synchronous with *.newsrc*
%./.rnhead	temporary header file to pass to a mailer or news poster
%./.rnmac	macro and keymap definitions
%p	your news save directory, usually ~/News
%x/active	the list of active newsgroups, usually /usr/lib/news/active
%P	the public news spool directory, usually /usr/spool/news
%X/INIT	system|wide default switches

SEE ALSO

newsrc(5), more(1), readnews(1), Pnews(1), Rnmail(1)

DIAGNOSTICS

Generally self|documenting, as they say.

BUGS

The **–h** switch can only hide header lines that *rn* knows about.

The '–' command doesn't cross newsgroup boundaries, and only undoes the last article selection.

If you edit your *.newsrc* while *rn* is running, *rn* will happily wipe out your changes when it decides to write out the *.newsrc* file.

Rn doesn't do certain things (like ordering articles on posting date) that the author feels should be handled by inews.

Marking of duplicate articles as read in crosslreferenced newsgroups will not work unless the Xref patch is installed in inews.

If you get carried away with % or escape substitutions, you can overflow buffers.

There should be no fixed limit on the number of newsgroups.

Some of the more esoteric features may be missing on machines with limited address space.

NAME

Pnews - a program for posting news articles

SYNOPSIS

Pnews newsgroup title
or
Pnews -h headerfile [oldarticle]
or
Pnews

DESCRIPTION

Pnews is a friendly interface for posting news articles. It will ask several questions, then allow you to enter your article, and then post it using the inews(1) program. If you type h and a carriage return at any point, *Pnews* will tell you what it wants to know.

The -h form is used when invoked from *rn*. If your editor can edit multiple files, and you want the article to which you are replying to show up as an alternate file, define the environment variable NEWSPOSTER as "Pnews -h %h %A". You can also modify the the NEWSHEADER environment variable to change the header file that *rn* passes to Pnews.

ENVIRONMENT

AUTHORCOPY
If defined, contains the name of a file to which the finished article will be appended.

Default: article not saved

DOTDIR
Where to find your dot files, if they aren't in your home directory. This is primarily for accounts which are shared by more than one person.

Default: $HOME

EDITOR The editor you want to use, if VISUAL is undefined.

Default: whatever your news administrator installed, usually vi.

HOME Your home directory.

Default: $LOGDIR

LOGDIR
Your home directory if HOME is undefined.

LOGNAME
Your login name, if USER is undefined.

Default: value of "whoami".

NAME Your full name.

Default: name from /etc/passwd, or ~/.fullname.

ORGANIZATION
Either the name of your organization, or the name of a file containing the name of your organization.

Default: whatever your news administrator chose.

USER Your login name.

Default: $LOGNAME

VISUAL The editor you want to use.

Default: $EDITOR

FILES

$DOTDIR/.article
~/dead.article

SEE ALSO

rn(1), Rnmail(1), inews(1)

DIAGNOSTICS

BUGS

Not the speediest program in the world, but maybe that's a blessing to the net.

Micro-emacs

micro-emacs est un éditeur de texte du domaine public. C'est un éditeur pleine page qui existe aussi sur l'Amiga. Vous trouverez ci-après, la liste **exhaustive** des commandes; mais seules 3 ou 4 d'entre elles sont véritablement d'un usage courant. Il s'agit de:

CTRL X CTRL S:	Sauvegarde du fichier édité
CTRL X CTRL C:	Sortie de l'éditeur (sans sauvegarde)
CTRL X CTRL I:	Insertion d'un fichier à l'endroit où se trouve le curseur
CTRL K	Suppression des caractères se trouvant à droite du curseur

COMMANDES de EE (micro-emacs)

abort-command	^G
add-mode	^XM
add-global-mode	M-M
apropos	
backward-character	^B
	M-D
begin-macro	^X(
beginning-of-file	M-<
beginning-of-line	^A
bind-to-key	M-K
buffer-position	^X=
case-region-lower	^X^L
case-region-upper	^X^U
case-word-capitalize	
case-word-lower	M-L
case-word-upper	M-U
change-file-name	^XN
change-screen-size	M-^S
change-screen-width	M-^T
clear-and-redraw	^L
clear-message-line	
copy-region	M-W
count-words	M-^C
ctlx-prefix	^X
delete-blank-lines	^X^O
delete-buffer	^XK
delete-mode	^X^M
delete-global-mode	M-^M
delete-next-character	^D
delete-next-word	
delete-other-windows	^X1
delete-previous-character	^H
delete-previous-word	M-^H
	M-
delete-window	^X0
describe-bindings	
describe-key	^X?
detab-line	^X^D
end-macro	^X)
end-of-file	M->
end-of-line	^E
entab-line	^X^E
exchange-point-and-mark	^X^X
execute-buffer	
execute-command-line	
execute-file	

execute-macro	^XE
execute-macro-1	
execute-macro-2	
execute-macro-3	
execute-macro-4	
execute-macro-5	
execute-macro-6	
execute-macro-7	
execute-macro-8	
execute-macro-9	
execute-macro-10	
execute-macro-11	
execute-macro-12	
execute-macro-13	
execute-macro-14	
execute-macro-15	
execute-macro-16	
execute-macro-17	
execute-macro-18	
execute-macro-19	
execute-macro-20	
execute-macro-21	
execute-macro-22	
execute-macro-23	
execute-macro-24	
execute-macro-25	
execute-macro-26	
execute-macro-27	
execute-macro-28	
execute-macro-29	
execute-macro-30	
execute-macro-31	
execute-macro-32	
execute-macro-33	
execute-macro-34	
execute-macro-35	
execute-macro-36	
execute-macro-37	
execute-macro-38	
execute-macro-39	
execute-macro-40	
execute-named-command	M-X
execute-procedure	M-^E
execute-program	^X$
exit-emacs	^X^C
fill-paragraph	M-Q
filter-buffer	^X#
find-file	^X^F

forward-character	^F
	M-C
goto-line	M-G
goto-matching-fence	M-X^
	^XZ
handle-tab	^I
hunt-forward	
hunt-backward	
help	M-?
i-shell	^XC
incremental-search	^XS
insert-file	^X^I
insert-space	^C
insert-string	
kill-paragraph	M-^W
kill-region	^W
kill-to-end-of-line	^K
list-buffers	^X^B
meta-prefix	^]
move-window-down	^X^N
move-window-up	^X^P
name-buffer	M-^N
newline	^M
newline-and-indent	^J
next-buffer	^XX
next-line	^N
	M-B
next-page	^V
	M-S
next-paragraph	M-N
next-window	^XO
next-word	M-F
nop	M-FNC
	M-FNR
open-line	^O
overwrite-string	
pipe-command	^X@
previous-line	^P
	M-A
previous-page	^Z
	M-V
	M-T
previous-paragraph	M-P
previous-window	^XP
previous-word	
query-replace-string	M-^R
quick-exit	M-Z
quote-character	^Q

read-file	^X^R
redraw-display	M-^L
	M-!
resize-window	^XW
restore-window	
replace-string	M-R
reverse-incremental-search	^XR
run	M-^E
save-file	^X^S
save-window	
scroll-next-up	M-^Z
scroll-next-down	M-^V
search-forward	^S
search-reverse	^R
select-buffer	^XB
set	^XA
set-encryption-key	M-E
set-fill-column	^XF
set-mark	M-
	M-.
shell-command	^X!
shrink-window	^X^Z
split-current-window	^X2
store-macro	
store-procedure	
transpose-characters	^T
trim-line	^X^T
unbind-key	M-^K
universal-argument	^U
unmark-buffer	M-~
update-screen	
view-file	^X^V
wrap-word	M-FNW
write-file	^X^W
write-message	
yank	^Y